

C64

3

**The
Book of
FACTS**

The information in this booklet will give you an insight into the powers of the Commodore. If you are a new computer owner you should find the color computing section very interesting. The more advanced user's will find the section on pokes and peeks very informative. Most examples require no previous computer experience. The function of this booklet is more of a reference guide, than a tutorial. Every program sample has a complete description of how it works. The contents within contains information on most of the important functions of the Commodore computer. Most of the material is written so that the user can start computing their very first day, without spending hours reading hard to understand computer manuals. There are tips that everyone can use in the everyday use of their computer. An explanation of each example is included to make it as clear as possible. Please refer to the index for help in finding the information that interests you the most.

INDEX:

* GETTING STARTED	PAGE 2
* KEYBOARD COLORS	PAGE 3
* ASCII COLOR VALUES	PAGE 5
* POKEING COLOR	PAGE 6
* PROGRAMS	PAGE 7
* SOUND	PAGE 10
* DISK & TAPE FUNCTIONS	PAGE 12
* DISK DRIVE INSTRUCTIONS	PAGE 13
* DIRECTORY STRUCTURE	PAGE 14
* DISK COMMANDS	PAGE 15
* DOS ERROR MESSAGES	PAGE 18
* POKE & PEEK	PAGE 19
* BASIC KEY WORDS	PAGE 23
* PROGRAM PROTECTION	PAGE 26
* DISK PROTECTION	PAGE 28
* ASCII CHARACTER CODES	PAGE 31
* SCREEN DISPLAY CODES	PAGE 34

The following pages will take you step by step through some simple program examples. After each example there is an explanation of what the program is doing. If you are new to computing, PLEASE DO NOT OVER LOOK THIS PAGE.

GETTING STARTED: It is important that you read and understand the following information as it will be used very frequently as you procede.

NEW - The NEW command is used to clear the computer memory of any basic program currently in memory. The purpose is to start each new program without the possibility of unwanted data from the previous program in memory, from becoming part of the new program. The NEW command should be used before typing in any any program.

CLEARING THE SCREEN - Holding the RUN/STOP key down and then pressing the RESTORE key will clear the screen. This will not destroy the program currently in memory, by typing the word LIST your program will be displayed again. You will find that this technique is useful when editing your programs. This procedure can be used to clear the display before using the command NEW.

CURSOR KEYS - The cursor keys are two keys on the lower right hand side of the keyboard. These keys are marked with the letters CRSR, and arrows pointing up/down on one and left/right on the other. The key on the left will move the cursor down when pressed, and up when used with the SHIFT key. The cursor key on the right will move the cursor right when pressed, and left when used with the SHIFT key. It is possible to move the cursor anywhere on the screen using these keys.

RETURN KEY - The RETURN key is used to enter data into memory. When you type in a command, such as the NEW command, the computer cannot respond until the command is entered by pressing the RETURN key. All data is entered in this manner. After typing in a program line, pressing the RETURN key will enter it into memory.

LINE NUMBERS - The computer has to read a sequential list of numbers that precede each program line. Without line numbers it would be impossible for the computer to follow the program. Line numbers are not required if you wish to enter commands that are to be executed immediately. NO line number can be higher than 63999.

KEYBOARD COLORS

The Commodore computer has a total of sixteen colors. This section will explain some uses for these colors. The examples shown can be modified for your own programs or just for fun. There are three different methods to access color in a basic program. The first of these methods, is direct keyboard input. Colors with values from 0 to 7 are accessed by holding the CTRL key down and any number key from 1 through 8. To access the values from 8 to 15 hold down the Commodore key (C=) and any number key from 1 through 8.

CHART ON KEYBOARD LOCATION OF COLORS

CTRL KEY	NUMBER KEY	- Color	Screen will display
Ctrl	1	- Black	
Ctrl	2	- White	
Ctrl	3	- Red	
Ctrl	4	- Cyan	See Page 36.
Ctrl	5	- Purple	
Ctrl	6	- Green	
Ctrl	7	- Blue	
Ctrl	8	- Yellow	

COMM KEY C=	NUMBER KEY	- Color	Screen will display
C=	1	- Orange	
C=	2	- Brown	
C=	3	- Light Red	
C=	4	- Dark Grey	See Page 36.
C=	5	- Medium Grey	
C=	6	- Light Green	
C=	7	- Light Blue	
C=	8	- Light Grey	

Please type in the program example below and run it. Between each letter there is a space, insert any color using the chart above in these spaces.

10 print "Y O U R N A M E" (press return key) then type the word
RUN (press return key).

The next example will be a display of colored bars. First an explanation of what reversed text is and how to turn it on. The flashing square displayed on the screen is called the cursor. The cursor is actually a matrix of pixels (small dots). The cursor has all of its pixels turned on. The make up of each letter, number, or character is produced, by turning on only those pixels that correspond to that particular character. When the text is reversed, only those pixels that surround a particular character are turned on. To turn on the reverse mode, hold the CTRL key down and press the number nine key, everything typed will be displayed in reversed text. To turn reversed mode off, hold the CTRL key down and press the 0 (zero) key. Please type in the following:

Hold the run/stop key down then hit the restore key to clear the screen.

STEP 1. Hold the CTRL key down then press the number nine key.

STEP 2. Type in any color using the chart on the previous page.

STEP 3. Hold the space bar down until the colored bar is the desired length.

STEP 4. Next type in a new color and repeat steps 2 and 3 until all sixteen colors have been displayed.

Please type in the following example of programing color from the keyboard. Each program line should be typed in then the RETURN key should be pressed to enter each line into memory.

```
10 A = (RND (0)*16 + 1:REM GENERATES RANDOM NUMBERS
20 X$ = MID$ ("SEE BELOW ",A,1):REM SEE EXPLANATION.
30 PRINT SPC(A( X$):REM SEE EXPLANATION BELOW
40 PRINT "YOUR NAME":PRINT:REM PRINTS NAME THAN ONE
   BLANK LINE
50 FOR D = 1 TO 50: NEXT D:REM COMPUTER STOPS TO COUNT
   TO 50
60 GOTO10:REM RETURN TO LINE 10
```

EXPLANATION: Line 10 creates random numbers from 1 to sixteen. Line 20 the variable X\$ equals the MID\$ values. Between the quote marks are the sixteen colors. The method to type them in follows: First hold the CTRL key down and press every number key from 1 through 8, this gives you the first eight colors. Next hold down the COMMODORE key and press the number keys 1 through 8 again this gives you all sixteen colors. The A after the last quote mark is the value generated by the RND statement in line 10. Starting from the far left side of the quote marks the value in line 10 counts left to right each color until the value is reached. Then the 1 picks one color. EXAMPLE: SUPPOSE THE VALUE GENERATED IN LINE 10 IS 7, THEN THE COLOR CHOSEN WOULD BE THE SEVENTH FROM THE LEFT (the color blue). Line 30: prints the number of spaces that are generated in line 10 then the color. Line 40 print your name in the color chosen in line 20 and one blank line. Line 50 displays line 40 to the count of 50. Line 60 starts the process over again until stopped by holding down the RUN/STOP key then hitting the RESTORE key.

COLOR USING ASCII:

The next method to program in color is the use of the ASCII values. The chart below shows the ASCII values for all sixteen colors.

COLOR	ASCII VALUE	COLOR	ASCII VALUE
Black	144	Orange	129
White	5	Brown	149
Red	28	Light Red	150
Cyan	159	Dark Grey	151
Purple	156	Medium Grey	152
Green	30	Light Green	153
Blue	31	Light Blue	154
Yellow	158	Light Grey	155
Reverse On	18	Reverse Off	146

The following program is an example of using ASCII values to program color. It is good programming practice to follow the steps below before starting to type in new program. Please type in the following:

STEP 1. Type the word NEW (press the return key).

STEP 2. Now hold the RUN/STOP key down and hit the RESTORE key to clear the screen.

```
10 PRINT CHR$(147) : REM CLEARS THE SCREEN
```

```
20 INPUT "CHOOSE ASCII COLOR VALUE";A : REM PLEASE DO  
NOT USE ANY NUMBER THAT IS NOT ON THE ASCII CHART
```

```
30 PRINT CHR$(18) CHR$(A);"    " : REM TURNS REVERSE ON  
THEN PRINTS TEN SPACES IN THE COLOR THAT IS INPUT  
IN LINE 20.
```

```
40 FOR A = 1 TO 1000 : NEXT A : REM THE COMPUTER COUNTS  
FROM 1 TO 1000 THEN CONTINUES
```

```
50 GOTO 20 : REM GOTO LINE 20
```

NOTE: Press the return key after typing each program line. After typing the complete program in, type the word RUN then press the return key. The word REM and any word that follows is an explanation of each line and has nothing to do with the program running, typing this in is optional. The last method of color programming consists of putting values into special memory locations of the computer that control the color. Before you continue to this last section of color computing, please take a short review.

- A. The Commodore has 16 colors.
- B. There are three methods to program with color:
 - 1. Direct from keyboard.
 - 2. ASCII values.
 - 3. Inserting color values in memory.
- C. The cursor is a matrix of pixels with the shape of a square.
- D. A pixel is a small dot.
- E. When a character is reversed all pixels that surround it are turned on.
- F. When typing in a new program it is good practice to follow these steps:
 - 1. Type in the word NEW (then press the return key).
 - 2. Hold the RUN/STOP key down and hit the restore key (will clear the screen.)

Inserting values into special memory locations may seem complex, but is very easy to understand. Below is a list of values for this purpose.

COLOR VALUES

Color	Value	Color	Value
Black	0	Orange	8
White	1	Brown	9
Red	2	Light Red	10
Cyan	3	Dark Grey	11
Purple	4	Medium Grey	12
Green	5	Light Green	13
Blue	6	Light Blue	14
Yellow	7	Light Grey	15

MEMORY LOCATIONS

- POKE 53280,? (screen border location)
- POKE 53281,? (screen background location)
- POKE 646,? (cursor location)

The question mark can be from zero to fifteen. Refer to the color chart above for the color you wish to input. As you can see there is the word POKE preceding each number, the reason for this is that, you must give the computer a command before it can perform any function. When you wish to insert a value into any memory location, that value has to be POKEd there. When you POKE a value in memory, it means the same as to insert it. Type in the following examples.

POKE 53280,0 (means place in memory location 53280 the value 0, which is the color black)

POKE 53281,1 (means place in memory location 53281 the value 1, which is the color white)

POKE 646,2 (means place in memory location 646 the value 2 which is the color red)

Type in the program below and after each line press the RETURN key to enter the program in memory. When all lines are typed in, type the word RUN and press the RETURN key.

```
10 PRINT "WHAT EVER YOU LIKE"
```

```
20 GOSUB 40:REM GOES TO LINE 40
```

```
30 GET$:IF A$=" " THEN 10:REM PLEASE SEE EXPLANATION BELOW
```

```
40 POKE646,N:N=N+1:IFN= 15 THEN N =1:REM PLEASE SEE EXPLANATION BELOW
```

```
50 RETURN:REM RETURNS TO LINE AFTER GOSUB (LINE 30)
```

EXPLANATION: Line 10 prints what ever is between the quote marks. Line 20 goes to line 40. Line 40 puts the value of N in the character color memory location then N is incremented by one. Then if the statement keeps the counter from going over the value 15 because there are no color codes that are above that. Line 50 goes to line 30 and line 30 scans the keyboard for a key to be pressed, if no key is pressed then the program goes to line 10 and prints what ever is between the quote marks in the color that was returned by line 40.

PROGRAMS THAT PRODUCE COLOR

Type in the following programs and after each line press the RETURN key to enter the line into memory. After all lines have been typed in type the word RUN and press the RETURN key.

```
10 PRINT CHR$(147):REM CLEARS THE SCREEN
```

```
20 INPUT "PLEASE CHOOSE A NUMBER FROM 0 TO 15";A:REM ASKS FOR A NUMBER
```

```
30 POKE53280,A:REM PUT IN MEMORY LOCATION 53280 THE NUMBER INPUT IN LINE 20
```

```
40 GOTO 10:REM START OVER AGAIN
```

EXPLANATION: Line 10 is the character code to clear the screen. Line 20 asks for a number to be input from 0 to 15. Line 30 puts the value of the color input in line 20 in the border color memory location. Line 40 sends the program back to line 10 to start the process all over again until the RUN/STOP and RESTORE keys are pressed.

```
10 PRINT CHR$(147)
```

```
20 INPUT BORDER COLOR CHOOSE A NUMBER 0 TO 15";A
```

```
30 POKE 53280,A:REM BORDER COLOR
```

```

40 FOR DELAY=1 TO 1500:NEXT DELAY:REM THE COMPUTER
COUNTS FROM 1 TO 1500 THEN CONTINUES
50 PRINT CHR$(147)
60 INPUT "BACKGROUND COLOR CHOOSE A NUMBER 0
TO 15"; B
70 POKE 53281,B:REM BACKGROUND COLOR
80 FOR DELAY=1 TO 1500: NEXT DELAY
90 GOTO 10:REM START OVER

```

EXPLANATION: Line 10 clears the screen then line 20 asks for the border color to be input. Line 30 puts the value that was input from line 20 in the color memory location for the border. Line 40 displays the border color in line 30 for a count of from 1 to 1500. Line 50 clears the screen then goes to line 60. Line 60 inputs a background color then goes to line 70 which displays the background color chosen in line 60 for a count of from 1 to 1500 that is in line 80. Line 90 goes to line 10 and starts all over again.

Please add or replace the following program lines to your last program.

```

90 INPUT "CURSOR COLOR CHOOSE NUMBER 0 TO 15";C
100 POKE 646,C:REM CURSOR COLOR
110 FOR DELAY=1 TO 1500: NEXT DELAY
120 GOTO 10:REM START OVER

```

EXPLANATION: Line 90 will replace the old line 90, the new line 90 will input a color value to be put in line 100 which displays the cursor color input from line 90. Line 110 displays the cursor for the county of from 1 to 1500 then goes to line 120 which returns to line 10 and starts all over again. You may want to save this program on disk or tape if so refer to the section on disk and tape functions. The basic function of the three most important color memory locations has been explained.

Please type in the following in what is called the immediate mode. To enter program line in memory, press the return key.

```
FOR A = 0 TO 1 STEP 0:POKE53280,4:POKE53280,6:NEXT
```

EXPLANATION - Using the variable name A, start count at 0, end the count at 1, counting in steps of 0. Put in memory location 53280 (border color) the color purple. The put in the same location the color yellow. The last instruction (next) continues to the next step, which will repeat the process over and over again. Hint: Holding the CTRL key down will slow the process down, and depressing the RUN/STOP key will halt execution.

Type each program line in then press the return key to enter it into memory. After all lines have been typed and entered into memory, type the word RUN then press the RETURN key to execute each program below.

```
5 PRINT CHR$(147):REM CLEAR SCREEN
10 POKE53280,0:POKE53281,6
15 PRINT "BYE! SEE YOU LATER"
20 for D= 1 to 1000: NEXT D: REM COMPUTER COUNTS TO 1000
30 POKE53265, PEEK(53265) AND239:REM WILL BLANK THE
SCREEN WITH THE BORDER COLOR
35 FOR D= 1 TO 1000:NEXT D
40 PRINT CHR$(147)
45 PRINT "HI! HERE I AM AGAIN:REM SEE EXPLANATION
BELOW
50 POKE53265, PEEK(53265) OR 16:REM TURN SCREEN BACK ON
```

EXPLANATION - Line 5 prints chr\$(147), which is the chr\$ code for clear the screen. Line 10 pokes color memory locations with the color values for a black border color and a blue background color. Line 15 prints what is between the quotes. Line 20 is a program delay used as a timer for the amount of time that line 15 will remain displayed on the screen. Line 30 pokes memory location 53265 (vic control register) with the value already there, then ands it with the value 239. This will cause the screen to change to the border color. Line 35 another delay that times the amount of time that the screen will remain blanked. Line 40 clears the screen of what was printed on line 15. Line 45 prints the reverse S which is the CLR/HOME key, the reverse Q is the down cursor key. What this does is print line 45 10 lines down from the home position of the cursor. Line 50 or's the memory location 53265 with itself and the value 16. This line turns the screen back on.

```
5 PRINT CHR$(147):REM CLEARS THE SCREEN
10 PRINT "THE CORRECT TIME PLEASE (00HR:00MIN:00SEC)"
20 INPUT TI$
30 PRINT CHR$(147)
40 PRINT "    THE TIME IS >>> ";TI$:GOTO40
```

EXPLANATION - Line 5 clears the screen. Line 10 prints what is between the quote marks. Line 20 inputs the ti\$ function (the jiffy clock). Line 30 clears the screen of what was printed on line 10. Line 40 is printed 14 lines down and 7 spaces over. Also the time is printed to the screen and the goto 40 repeats the instructions on line 40 over and over until the RUN/STOP key is held down while pressing the RESTORE key stop the program execution.

SOUND

Programing sound on the Commodore can be very complicated. Normally it is beyond the ability of most beginning computer owners to produce sound from the computer that would be the quality necessary to perform music. Please do not let this discourage those of you that wish to program your favorite song. There are many commercial programs on the market that can produce excellent music or sound synthesis you may wish to experiment with. Below you will see a program example, this program produces various types of sound. After typing on each line, press the RETURN key, to enter the line into computer memory. When you are finished entering the last program line, type the word RUN and press the RETURN key to execute the program.

```
10 PRINT CHR$(147):REM CLEAR THE SCREEN
20 INPUT "SET VOLUME FROM LOWEST 0 TO HIGHEST 15";A:REM
  SET VOLUME OF SOUND
30 PRINT "TRIANGLE:SAWTOOTH:PULSE:NOISE"
40 PRINT " 17    33    65   129"
50 INPUT "SET WAVE FORM 17-33-65-129";B:REM CHOOSE FROM
  THE FOUR TYPES ON LINE 30
60 INPUT "SET FREQUENCY CONTROL 0 TO 127";C:REM SET SOUND
  FREQUENCY
70 POKE5496,A:REM MEMORY LOCATION FOR SOUND VOLUME
80 POKE54278,240:REM MEMORY LOCATION TO SUSTAIN THE
  LENGTH OF TIME THE SOUND WILL BE HEARD
90 POKE54275,2:REM MEMORY LOCATION FOR THE PULSE WAVE
  FORM WIDTH
100 POKE54276,B:REM MEMORY LOCATION THAT SELECTS TYPE OF
  WAVEFORM
110 FOR F=1 TO C
120 POKE54273,F+F:NEXT F:REM MEMORY LOCATION THAT
  CONTROLS SOUND FREQUENCY
130 GETA$:IFA1+""THEN 110
140 IFA$="R" THEN 250:REM IF THE R KEY IS DEPRESSED THAN
  GOTO LINE 250
150 IFA$="Q" THEN 350:REM IF THE Q KEY IS DEPRESSED THEN
  GOTO LINE 350
160 S=54272:REM START OF SOUND MEMORY LOCATIONS
170 FOR I=0 TO 28
180 POKE S+1,0:REM PLACE THE VALUE ZERO IN THE MEMORY
  LOCATION 54272
190 NEXT I
200 RETURN
```

```

250 GOSUB 160:PRINT CHR$(147):GOTO10
350 GOSUB 160:PRINT
CHR$(147):POKE631,82:POKE632,85:POKE633,78:POKE198,3

```

EXPLANATION - First line 10 clears the screen, then line 20 asks for a value from 0 to 15. This value sets the volume for the sound that is produced. LINE 30 through 50 asks for one of the four values that represent the type of wave form to be used. Line 60 asks for a value from 0 to 127 that controls the sound frequency. Line 70 puts the value input in line 20 into the memory location that controls the volume. Line 80 puts the value 200 in the memory location that controls the amount of time that the sound is sustained. Line 90 puts the value 2 in the memory location that controls the pulse wave form width. Line 100 puts the value input in line 50 in the memory location that controls which of the four types of wave forms will be produced. Line 110 to 120 is a FOR NEXT loop that increments by one the value input in the memory location that controls the frequency of the sound produced. This value will be increased each time it goes through the loop, depending on the value that is input in line 60. Line 130 uses the GET statement to search the keyboard for a key to be depressed, if no key is pressed then line 110 is executed again. The sound will continue to repeat until the R in line 140 is pressed or the Q in line 150 is pressed. If the letter R is pressed, line 140 send the program to line 250. The GOSUB statement on line 250 has the program execute lines 160 through 200. This puts a value of zero in all of the sound memory locations to reset them to normal. The the RETURN statement on line 200 returns the program to execute the rest of line 250, which clears the screen and then starts the program over again. If the letter Q is pressed, line 150 has the program go to line 350. Line 350 to execute the next statement. Chr\$(147) clears the screen then the letters R U N are poked into memory locations 631 through 633, which are the locations that temporarily store characters typed in from the keyboard. After the characters R U N are put into memory, the value three is put in the memory location that controls the number of characters in the keyboard buffer. The purpose is to print the word RUN on the same line as the cursor is on, so that when the RETURN key is pressed the program will execute again. This is only put in the program as an example of what can be done with these memory locations and has nothing to do with producing sound.

NOTE: They excellent sound capabilities of the Commodore are made possible by the SID chip. The Commodore programmer's reference guide will give a more detailed description of programming the SID chip.

DISK AND TAPE FUNCTIONS

There are two types of data storage for the Commodore, they are tape and disk. The cassette tape is the least expensive of the two. There are a number of manufactures that market a tape recorder or disk drive for the Commodore computer. The following list below will name a few:

CASSETTE TAPE

Commodore
Cardco

DISK DRIVE

Commodore
MSD
Indus GT

The tape recorder is a sequential device. It is only possible to read the tape from beginning to end. It does not have the ability to go backwards or skip around. There also is no easy way to find out what is on the tape without reading the entire length. The disk drive is a random access device. It has the ability to ready any part of the disk without reading from the beginning first.

THE INSTRUCTIONS BELOW ARE FOR TAPE (please read them carefully).

To hook up the tape recorder is quite simple. Take the cable on the back of the tape recorder and insert it in the back of the computer, right behind the 4, 5, and 6 keys. Please make sure that you line up the teeth correctly, do not use excessive force. Now the tape recorder is ready to use. You should avoid using tapes that are over thirty minutes as they have a tendency to stretch, five to ten minute tapes are preferred.

To load from tape means that you are transferring data from tape to the computer. After a load is completed you will see a ready prompt, after which you type in (run) and then your program will execute.

STEP 1. To load a program from tape first you must have a program on it. We will assume that a program is on the tape, first rewind the tape and reset the counter so that you have a reference point. Now type in the following: LOAD "PROGRAM NAME",1 (then press the return key) (note the one is the device number for the recorder and is optional).

STEP 2. Press the play button on your recorder (the computer screen will prompt you to do-so(. The screen will go blank, and when it reappears you will see the message below:

SEARCHING FOR (program name)

FOUND (program name)

Press the Commodore key, lower left hand corner of the keyboard. The screen will go blank a short time then it will display the message LOADING.

STEP 3. After the program has loaded the ready prompt will be displayed. You should now enter the word (run), to execute the program.

To save to tape means that you are transferring data to the tape.

STEP 1. To save a program first we have to make one, please type in the program below:

Type in the word (new) then press the return key.

10 PRINT "ANYTHING YOU LIKE" (press return key)

20 END (press return key)

STEP 2. Now that we have a program lets save it to tape. Please type in the following:

SAVE "PROGRAM NAME", 1 (press return key) (note the one is the device number for the recorder and is optional)

STEP 3. The screen will prompt you to press the play and record keys simultaneously, after the save is completed the ready prompt should reappear.

You can load and run a program by pressing the shift and run/stop keys simultaneously. Always remember to rewind the tape before any load is to be executed. If load is typed by itself, then the first program found on the tape will be loaded. If you were to simply type save, then a program would be stored on the tape with no file name. There is the verify command, which will verify that a saved program has been store properly. The verify command is executed by typeing VERIFY "PROGRAM NAME" and pressing the return key. If a verify error is displayed then the program has not been stored properly. Verify can also be used to find the end of a program, so that you may save a new program at the end of the proceeding program. Please follow the example below:

STEP 1. When you have a new program that is to be put on a tape with others type in VERIFY "PROGRAM NAME" (press return key). (Note use the name of the last program on the tape).

STEP 2. While searching to verify the last program all others will be bypassed. A verify error will be displayed because the last program is not the same as the one in the computers memory, but you will be at the end of the last program on tape.

STEP 3. Now save your new program at this location on the tape.

THE INSTRUCTIONS FOR THE DISK DRIVE

To hook up the disk drive is as simple as the tape recorder. The back of the disk drive has three sockets one at the bottom next to the on/off switch and two identical sockets at the right hand corner. The socket by the switch is the power cord and should be plugged into the same outlet as the computer. Take the black cord and plug one end in the back of the computer between the seven and eight keys, and plug the other end in either of the two upper right hand sockets. Because the disk drive is used more often than the tape recorder, it will be explained in much more detail. The disk that is inserted in the disk drive is where the data is stored. Data is stored in the form of blocks which hold up to 256 bytes information. There are 683 blocks on each diskette. The blocks are stored on tracks, there are 35 tracks on the Commodore disk. The drive has a program called the DOS (note disk operating system) this program keeps track of the blocks. It organized into a BAM (note - block allocation map) and a directory. The BAM and directory is stored on track 18 of the disk. When ever a program is saved or a data file closed, on the disk, the BAM is updated to show how many blocks are available. All data stored is arranged in files, there are five types of files. The following is a list and explanation of the five types of files.

PRG
REL
SEQ
USR
DEL

RELATIVE: A relative file is used to store data, all data records are numbered. This type of file is tougher to work with, but is used best if there are a lot of files. Relative files are accessed with the open command, but before reading this type of file you must set a pointer to zero in on a particular record. The relative file is faster to access data than a sequential file because of this.

SEQUENTIAL: A sequential file is something like a relative file, except all data has to be read from beginning to end. Because of this it is slower to use and takes up more computer memory. The sequential file is easy to use because you are not required to set a pointer to any particular record.

USER: A user file is a very special type of file. This type of file is used mostly as a form of protection. The user file can be treated as it were a sequential file. This type of file will be erased when the validate command is executed. Programs that are wrote for the disk operating system will most likely have data stored in user files.

DELETED: The deleted file is no longer stored on the directory and no blocks are allocated for it in the barn. This file still exists on the disk until the blocks formerly held by it are reallocated to a new file. When a file is scratched from the disk it becomes a deleted file.

Below is what a typical directory looks like:

"PICKUP [REDACTED]"

14

The format can be abbreviated as follows:

OPEN15,8,15,"N:NAME,ID" (press return key)

After disk is formatted the FILE that was open must be closed, using this statement: **CLOSE15** (press return key)..

HINT - If you already have a formatted disk and wish to reformat it. There is a faster method than the standard format it is as follows:

OPEN15,8,15,"S:NAME" (press return key)

SCRATCH A FILE:

EXPLANATION - When a file is scratched from the disk it is erased. The procedure is the same as to format, except that an S (scratch) is used instead of the N (new), and the ID is not used. It is possible to scratch more than one file by using pattern matching, for example if there are two files and their names are test and track. You would type **OPEN 15,8,15,"S:T"**, both programs would be erased. Another example of this technique, used on three files with the names, rats, bats, and cats would be as follows:

OPEN15,8,15,"S:?ATS", this would erase all three files.

RENAME A FILE:

OPEN15,8,15,"R:NEW NAME = OLD NAME" (press return key)
(the rename command will now work with files that are open)

EXPLANATION - This is very simple, the R stands for rename. If you have a file with the name planes and you want to change the name to airplanes, you would type the following, **OPEN15,8,15,"R: AIRPLANES= PLANES.** (press return key).

VALIDATE A DISK:

OPEN15,8,15,"V" (press return key)

EXPLANATION - When a disk has been used over a period of time, the directory can become disorganized. Usually this is caused by repeated saving and scratching of files. This usually results in one or more blocks on the disk that cannot be used because they are too small. The validate command will reorganize the disk to make the most out of the space that is available.

INITIALIZE THE DRIVE:

OPEN15,8,15,"I" (press return key)

EXPLANATION - There will be times when the disk drive will not respond to your commands. This is caused by an error condition. When the drive is initialized it will be returned to the same state as when first turned on.

READING THE ERROR CHANNEL:

10 OPEN15,8,15

10 INPUT "15,A\$,B\$,C\$,D\$"

30 PRINT A\$,B\$,C\$,D\$

EXPLANATION - The basic routine above will read the error channel. The purpose of reading the error channel, is to find out what type of error condition exists, if the red light on the drive starts to blink and the drive motor stops. Load and run the program above. The following is

an explanation of the program above: After opening a channel, 4 variables are read and describes the error condition. The first variable is the error number, the second is the error description, the third is the track number on which the error occurred, and the fourth is the block number of the track. Lastly the 4 variables are displayed on the screen.

VERIFY:

OPEN15,8,15,"V" (CHECKS CURRENT MEMORY AGAINST PREVIOUSLY SAVED FILE ON DISK)

LOAD PROGRAM FILE:

LOAD "PROGRAM" NAME, 8 (BASIC PROGRAMS)

LOAD "PROGRAM NAME",8,1 (MACHINE LANGUAGE PROGRAMS)

LOAD DIRECTORY:

LOAD "\$",8 (LOADS A DIRECTORY)

LOAD "\$",8 (LOADS THE DIRECTORY BUT ONLY LIST THE DISK NAME AND BLOCKS FREE)

LOAD "\$*=P",8 (LOADS DIRECTORY AND LISTS ONLY PROGRAM FILES)

(SUBSTITUTE THE P FOR S (SEQ.) OR R (REL) FILES)

SAVE PROGRAM FILE:

SAVE "PROGRAM NAME",8 (ONLY BASIC PROGRAMS CAN BE LOADED IN THIS MANNER)

SAVE AND REPLACE:

OPEN15,8,15"@0:PROGRAM NAME" (REPLACES A PROGRAM WITH THE SAME NAME WITH ONE IN MEMORY)

List Command:

List 100 (List line 100 only)

List 100 (Lists line 100 through end of program)

List 100-200 (Lists line 100 through 200)

List 100 (List from start of program to 100)

SUMMARY OF DISK OPERATING SYSTEM [DOS] error messages

NOTE: At times you will encounter error messages displayed on the screen while using the disk drive. The list below, gives the number and type of error.

0	OK (no error exists)
1	Files scratched response (not an error condition)
2-19	Unused error messages (please ignore)
20	Block header not found on disk
21	Sync character not found
22	Data block not present
23	Checksum error in data
24	Byte decoding error
25	Write-verify error
26	Attempt to write with write protect on
27	Checksum error in header
28	Data extends into next block
29	Disk ID mismatch
30	General syntax error
31	Invalid command
32	Long line
33	Invalid filename
34	No file given
39	Command file not found
50	Record not present
51	Overflow in record
52	File too large
60	File open for write
61	File not open
62	File not found
63	File exists
64	File type mismatch
65	No block
66	Illegal track or sector
67	Illegal system track or sector
70	No channels available
71	Directory error
72	Disk full or directory full
73	Power up message, or write attempt with DOS mismatch
74	Drive not ready

For a complete description of the DOS error messages read pages 53-56 of your disk drive user's guide.

POKE AND PEEK

POKE (MEMORY LOCATION), (VALUE)

EXPLANATION - The poke command inserts a value in a decimal memory location. You will be unable to change the value of memory locations in ROM (read only memory), it is only possible to change values in RAM (random access memory). The reason is ROM can only be read from not wrote to. This command will be used in various ways, one of which is to put a color value in memory. An example would be as follows:

POKE 53280, (memory location for the border color) 1 (value for the color white), the result will be displayed on the screen as a white border. The memory location can be from 0 through 65535, the value can be from 0 to 255.

PEEK (VALUE)

EXPLANATION- The peek command reads from a memory location. This command is usually used in conjunction with the poke statement, but is not required. An example of the peek command is as follows: **PEEK (204)**, will return a value of 0. This is the memory location for the cursor blink toggle, 0 = cursor on - 1 = cursor off.

USES FOR POKE AND PEEK: Data storage, Sound generation, Loading assembly language routines in memory, Screen graphic displays, Modify the operating system parameters.

MORE EXAMPLES: NOTE - Always use the decimal number when using the poke statement.

A. POKE 144,0 (144 is the location that contains the kernal input/output status word value. Specific values will perform, varied status functions. If the status gets misdirected, the computer may respond with the device not present message. The normal value for this location is 0, poking this value in location 144 should fix this problem.)

B. POKE 145,127 (145 is the memory location than contains the status flig for the STOP and RVS keys. The normal value for this location is 255. A value of 127 will act the same as the RUN/STOP key.)

C. POKE 160-162 (Locations 160 through 162 are the memory locations for the jiffy clock. These locations act as a timer for the computer. A jiffy is 1/60 of a second. These locations will reset to zero by poking a value of zero in all three locations.)

D. POKE 198,0 (198 is the memory location for the number of characters in the keyboard buffer. The normal value for this location is zero. It is possible to clear the buffer of any characters that it contains, by poking a value of zero here.)

E. POKE 199,18 (199 is the memory location that controls the reverse character switch. The normal value for this is 0. This value changes to 18 when the reverse character mode is used (holding the CTRL key down and pressing the nine key). It is possible to use this poke in a program to turn on reverse characters, but it most likely slower than using the RVS on and off keys.)

F. POKE 209-210 (209-210 are the memory locations of the pointer for the current screen line address. Different values for these locations will

position the word output to the screen. Type and run the following program for an example of this:

```
10 POKE209,20
```

```
20POKE210,7
```

```
30 PRINT "COMPUTER"
```

G. POKE 211 (211 is the location that stores the number of spaces that are used on a printed line. A value of 10 is equivalent to ten spaces. An example of this is in the program below.)

```
10 POKE211,10
```

```
20 PRINT "COMPUTER"
```

H. POKE 212 (212 is the quote mode flag, a value of zero in this location will take you out of quote mode.)

I. POKE 649,0 (649 is the memory location for the keyboard buffer. The normal value for this location is 10. Using different values will change the number of characters the buffer can hold before it loses characters. Putting a zero in this location will disable the keyboard. If you hold the RUN/STOP down and press the RESTORE key, you will regain control.

J. POKE 650,255 (650 is the memory location that controls the repeating of certain keys. When the computer is first powered up, the value in memory location 650 is 0. Only the space bar, cursor keys, and the insert/delete key will repeat. Any value from 64 to 127 will stop all keys from repeating. Values from 128 to 255 will make all keys repeat.)

K. POKE 657, 128 (657 is the memory location that controls the function of switching from upper case to lower case. The normal value for this location is 0. To disable shifting from upper case to lower case, use the value 128.)

L. POKE 755,200 (775 is the list vector, the normal value of this location is 167. A value of 200 will disable the list function.)

M. POKE 788,52 (788 is the interrupt request vector, value of 52 will disable the RUN/STOP key and also the jiffy clock. A value of 59 will restore both.)

N. POKE 808,239 (808 is the kernal stop vector. The normal value for this location is 237. Putting the value 239 will disable the RUN/STOP key without stopping the clock.)

O. POKE 808,225 (Disables the RUN/STOP and RESTORE keys.

P. POKE 1024-2023 (These are the location where data is stored to be seen on the screen. To put a diamond in the middle of the screen you would poke 1524,90. The screen has 1000 locations.)

Q. POKE 55296-56295 (These are the matching color locations, you use with the screen locatins. If you poke, 55756,2 our diamond will turn red.)

R. POKE 53272,23 (53272 is the memory location that controls the screen and character memory. A value of 23 will display in lower case characters. A value of 21 will convert it back to upper case.

S. POKE 53280,8 (53280 is the memory location for the screen border color The values for this location are from 0 to 15. The value 8 will change the border color to orange.)

T. POKE 53281,9 (53281 is the memory location for the screen

background color. The values for this memory location are from 0 to 15. The value 9 will change the screen background to brown.)

U. POKE 646,2 (646 is the memory location for the cursor color. The values for this location are from 0 to 15. A value of 2 will change the cursor color to red.)

V. POKE 56341,0 (56341 is a memory location that will control the speed at which the cursor will blink. The normal value for this location is 57. A value of 0 will cause the cursor to blink the fastest, and a value of 255 is the slowest.)

IMPORTANT MEMORY LOCATIONS:

1. 43-44 (These addresses point to the start of basic text. This is where a basic program begins. Basic text normally starts at \$0801 (hexidecimal) - 2049 (decimal). The hexidecimal and decimal answers are both the same location, but are calculated differently. Decimal numbers are counted in tens, hexidecimal numbers are counted in sixteenths.)

EXAMPLE: (1, 2, 3, 4, -5, 6, 7, 8, 9, A, b, C, D, E, F, 10)

NOTE: ALWAYS USE DEGIMAL NUMBERS FOR VALUES WHEN USING POKE AND PEEK FUNCTIONS.

2. 55-56 (These addresses are the pointers for the highest memory location used by a basic program.)

3. 57-58 (These addresses are the location where the current basic line number is stored.)

4. 59-60 (These addresses are the location where the previous basic line number is stored.)

5. 115-138 (These locations are a portion of the computers operating system, and since they are in RAM (random access memory,) they can be modified by the user.)

6. 152 (This is the location that contains the number of open files, no more than ten files can be opened at one time.

7. 153 (This is the location that stores the input device default number, which is 0 (keyboard.)

8. 154 (this is the location that stores the CMD output device default number, which is 3 (screen).

9. 186 (This location contains the value of the key, most recently pressed.)

10. 197 (this location contains the value of the key, most recently pressed.)

11. 206 (This location ocntains the value for the character under the cursor, the normal value is 32 (space).

12. 213 (This location contains the value for the lenth of screen line, the normal value is 39.)

13. 214 (This location contains the value for the vertical screen location of the cursor.)

14.631-640 (These locations temporarily store characters typed from

the keyboard. NOTE: Used in conjunction with location 198, it is possible to enter characters the same as if they were typed from the keyboard.) EXAMPLE: (POKE631,65:POKE198,1 - this will print the letter A on the screen.)

15. 646 (This location contains the character color value.)

16. 647 (this location contains the value of the background color under the cursor.)

17. 652 (This location stores the value for the key repeat delay counter.)

18. 653 (The value in this location corresponds to the following three keys, SHIFT, CTRL, and the Commodore. The value returned when the pressing the SHIFT key is 1, the value for the CTRL key is 4, and the value for the COMMODORE key is 2. If all three are pressed simultaneously the value is 7. There are many uses for this address one example would be the following: WAIT 653,1,0 - the program will stop until the SHIFT key is depressed, the 0 after the 1 resets location 653.)

19. 828-1019 (These locations are the buffer for the tape, this is the place where data is input first when loading from tape.)

The memory locations listed, will prove to be a valuable resource, when programing or just exploring the power of the Commodore computer. Please feel free to make use of these memory locations and their functions. The computer cannot be harmed by modifying the contents of these memory locations. The pages of this book contains many program examples of these memory locations, using a variety of techniques. Below is a list of some useful routines that can be executed using the system statement.

MEMORY LOCATIONS OF ROUTINES: NOTE (The word SYS before each memory location, is the statement that will execute each routine.)

1. SYS 58692 (Clears screen and homes the cursor.)

2. SYS 64738 (Cold start reset, with screen power up message.)

3. SYS 64760 (Faster cold start reset, with screen power up message.)

4. SYS 64767 (Reset with no screen color change.)

5. SYS 51712 (Warm start)

BASIC KEY WORDS

Command	Abbreviation	Definition
ABS	A(SHIFTED B) -	A numeric function that returns the absolute value of a number.
AND	A(SHIFTED N)	Can be a logical operator that checks the truth between two expressions.
ASC	A SHIFTED S)	A numeric function that gives the ASCII value of the first character in a string.
ATN	A(SHIFTED T)	A mathematical function.
CHR\$	A(SHIFTED H)	A string function that changes ASCII to a numeric value.
CLOSE	CL(SHIFTED O)	A input/output statement that closes a channel to a device.
CLR	(C(SHIFTED L)	A statement that resets all variables.
CMD	C(SHIFTED M)	A input/output statement that outputs data to an opened device or file that corresponds to a file number.
CONT	C(SHIFTED O)	A command statement that when used after a stop or end statement, will restart the flow of the program from that point.
COS	NONE	A mathematical function.
DATA	D(SHIFTED A)	String or numeric statements that are used in conjunction with the read statement.
DEF FN	D(SHIFTED E)	A statement that is a used defined substitute.
DIM	D(SHIFTED I)	A statement that defines the maximum range of an array or group of variables.
END	E(SHIFTED N)	A statement that stops a program from executing.
EXP	E(SHIFTED X)	A mathematical function
FOR	F(SHIFTED O)	A statement that sets up a variable as a counter, that is used in a for next loop.
FRE	F(SHIFTED R)	A statement that gives the amount of memory that is free.
GET	G(SHIFTED E)	A statement that scans the keyboard for a key to be key to be typed.
GET#	NONE	A input/output statement that will input a character from a device or file.
GOSUB	GO(SHIFT S)	A statement that will jump to a routine within a program, used with the RETURN statement.
GOTO	G(SHIFTED O)	A statement that will jump to a line within a program. (EXAMPLE: GOTO 10)
IF THEN	NONE	A statement that sets up a conditional branch. (EXAMPLE: IF A\$ = "Y" THEN SYS64738 (warm start)).
INPUT	NONE	A statement that halts program execution, waiting for user input. (EXAMPLE: INPUT "WHAT IS YOUR NAME"; A\$)
INPUT#	I(SHIFTED N)	A input/output statement that inputs data from a device or file.
INT	NONE	Returns the interger value of a number.
LEFT\$	LE(SHIFTED F)	A string function that returns a set number of characters from the left most end of a string, depending on the value given. (EXAMPLE: A\$ = "UPDOWN":PRINT LEFT\$ (A\$,2) display the word UP on the screen)

LEN	NONE	A interger function that counts the amount of characters in a specified string. (EXAMPLE: A\$ = "COMPUTER":PRINT LEN (A\$)-the result will be 8)
LJST	L(SHIFTED I)	A command that lists a program in current memory.
LOAD	L(SHIFTED O)	A command that loads a program from tape or disk.
LOG	NONE	A mathematical function.
MID\$	MSHIFTED I)	A string function that returns the part of a string that is set by the first value, counting from the far left, the length of the second value. (EXAMPLE: A\$ = "AROUNDTHEWORLD":PRINT MID\$(A\$,10,5)-the result of this would be the word WORLD.)
NEW	NONE	a command that when used will erase a program currently in memory.
NEXT	N(SHIFTED E)	A statement that is used in conjunction with the FOR statement, and serves as a counter that is incremented by 1, and tested for the end-value, to check to see if its time to stop the loop. (EXAMPLE: 10 FOR A = 1 TO 1000: NEXT A-this will count 1000 times before ending)
NOT	N(SHIFTED O)	A logical operator that can be used to compare a set of values or variables, for a negative result. (EXAMPLE: IF NOT X = Y THEN END- the program will end if the value of x does not equal the value of y.)
ON	NONE	A statement that is used with the goto and gosub statements. Depending on the value of a set of variables in a program, the ON statement will jump to any one of several predefined line numbers within the program. The ON statement can also be used inplace of the IF THEN statement. (EXAMPLE: ON A GOTO 10,20,30)
OPEN	O(SHIFTED P)	A command that will open a channel to a device or file.
OR	NONE	A logical operator that can compare a set of values for a true of false result. (EXAMPLE: IF A =50 OR B = 100 THEN END- this means if A and B are true then end the program)
PEEK	P(SHIFTED E)	Reads and returns the contents of a specified memory location. (EXAMPLE: POKE 53280,1-put in location 53280 the value
POKE	P(SHIFTED O)	Put a value in a specified memory location. (EXAMPLE: POKE 53280,1-put in location 53280 the value for color 1)
POS	NONE	Returns the current cursor position. (EXAMPLE: 10 IF POS(0) = 10 THEN END-when the cousor reaches position 20 the program will end)
PRINT	?	A statement that is used to output data to the screen. This data can also be directed to the printer by using the CMD statement. (EXAMPLE: PRINT "COMPUTER")
PRINT#	P(SHIFTED R)	A input/output statement that is used to output data to a device of file. (EXAMPLE: OPEN1,4:PRINT#1, "COMPUTER":CLOSE1-results in printing the word COMPUTER to the printer)
READ	R(SHIFTED E)	A statement that reads from a data statement. 1 READ A\$

EXAMPLE — 2 DATA "DISK DRIVE"

REM NONE A statement that cannot be executed, but serves as vehicle to store remarks in a program. (EXAMPLE: 1 PRINT "PRINTER":REM THIS IS LINE 1)

RESTORE RE(SHIFTED S) A statement that resets the pointer to the first data statement in the program. This can be used to read the same data as manytimes as needed. (EXAMPLE: 1 READ A\$ - reads the word TAPE twice)
 2 RESTORE
 3 DATA"TAPE"

RETURN RE(SHIFTED T) A statement that is used to return to the data or line number following the gosub statement.
 (EXAMPLE: 1 GOSUB 3 - goes to line 3 then returns
 2 END — end on line 2)
 3 RETURN

RIGHT\$ R(SHIFTED I) A string function that returns a set number of characters from the right most end of a string, depending on the value given.
 (EXAMPLE: A\$ = "SCREENCOLOR":PRINT RIGHT\$(A\$,5)-this will print the word COLOR)

RND R(SHIFTED N) A floating point function that creates a random number. (EXAMPLE: PRINT IRND(0)*10-this will print a number from 0 to 9)

RUN R(SHIFTED U) A command that will execute a program that is in memory.

SAVE S(SHIFTED A) A command that will store a program to tape or disk.

SGN S(SHIFTED G) A INTERIOR FUNCTION

SIN S(SHIFTED I) A FLOATING POINT FUNCTION.

SPC(S(SHIFTED P) A function that will print a specified number of spaces across the screen. (EXAMPLE: PRINT SPC910)-will print 10 spaces.

SQR S(SHIFTED Q) A floting point function that will return the square root of a specified value.

STATUS ST A function that gives the condition of any input or output operation.

STEP ST(SHIFTED E) A statement that works with the IF THEN statement. The STEP statement will increment the IF THEN function with a specified value.
 (EXAMPLE: FOR A = 1 TO 10 STEP 5-will loop twice)

STOP S(SHIFTED T) A statement that will stop the execution of a program. and print the line number from which STOP statement was executed.

STR\$ ST(SHIFTED R) A string function that returns a string variable from a numeric variable. (EXAMPLE: T=4:T\$=STR\$(T)A\$="THE TIME IS "+T\$+":00":PRINT A\$-will print (THE TIME IS 4:00)

SYS S(SHIFTED Y) A statement that will execute a memory location containing a machine language program or a kernal routine.
 (EXAMPLE: SYS 64738- will give a system cold start)

TAB(T(SHITED A) a function that acts like th SPC function, the TAB function should only be used with the PRINT statement, because it has no effect when used other-wise.

TAN NONE A MATHEMATICAL FUNCTION.

TIME **TI** A function that reads the internal timer of the computer also known as the jiffy clock. —(EXAMPLE: PRINT TI/30-this will display how many 1/30 second intervals have elapsed since power up)

TIME\$ **TI\$** A function that will return a six character time display. (EXAMPLE: 1 TI\$ = "000000":FOR T = 1 TO 10000: NEXT T:PRINT TI\$-will display 000010)

USR **U(SHIFTED S)** A function that accesses a user callable machine language routine. Memory locations 785 and 786 must contain the pointers for the start of the machine language routine.

VAL **V(SHIFTED A)** A function that converts a string variable to a numeric. (EXAMPLE: A\$ = "55": PRINT VAL (A\$)-will print a 55)

VERIFY **V(SHIFTED E)** A command that is used to compare a program currently saved to tape or disk with the same program in memory. (EXAMPLE: SAVE"TEST",8:VERIFY"TEST",8-will save and verify the program called test)

WAIT **W(SHIFTED A)** A statement that stops the execution of a program until the specified condition is met. (EXAMPLE: WAIT 198,1-will stop until any key is pressed)

FORMS OF PROGRAM PROTECTION:

There are many types of protection that can be used on software. Protection of software may be new to some of you, I will explain a few simple techniques. **FIRST, WHAT IS PROTECTION?:** Most commercial software uses some form of protection, that inhibit the user from making a copy or modification of their program. The most common forms of protection are listed below, with an explanation of what they do. There are so many types of protection, that it would be impossible to cover them all in just a few pages.

TYPES OF PROGRAM PROTECTION:

1. UNLISTABLE PROGRAM LINES
2. INVISIBLE PROGRAM LINES
3. UNLISTABLE PROGRAM
4. DISABLE KEYBOARD

TYPES OF DISK PROTECTION:

1. LOOP IN DISK DIRECTORY
2. UNLISTABLE DIRECTORY
3. INTENTIONAL DISK ERRORS

PROGRAM PROTECTION:

UNABLE TO LIST:

```
10 ?"HI THERE":REML
20 ?"THIS WILL NOT BE LISTED":REML
30 ?"NOR THIS":REML
40 END
```

EXPLANATION - When you try to list this program, you will get a syntax error. If you will notice the rem statement at the end of each line, has a graphic character after it. To produce this character hold down the SHIFT key and press the L key. The basic interpreter cannot understand what the shifted L character does, so it will return a syntax error. It is easy to overcome this problem by listing each line separately or moving the cursor to the line that create the syntax error, and pressing the return key. Now the line will be returned to normal. NOTE-THE QUESTION MARK IS SHORTHAND FOR THE PRINT STATEMENT.

INVISIBLE PROGRAM LINES:

```
10 POKE 775,200:PRINT"
20 ?"THE LINE ABOVE WILL BE HARD TO SEE"
30 ?"ALSO IF THIS PROGRAM IS RUN, THE LISTING WILL BE UNREADABLE."
40 END
```

The program above has two types of protection in line 10. The first is a poke statement that inserts a value of 200 in the list vector, disabling the list function, and second makes line 10 appear invisible. After the poke statement, there is a colon, and then a print statement. After that there is a quotation mark than a space, following is the protection. Put another quotation mark after the space, then press the delete key to erase it. Now hold the SHIFT key down and then the DELETE key (marked INST/DEL) down for a few seconds. You should see the cursor blinking fast, if you do not, retype the line 10 until you do. After you get the blinking cursor, release the SHIFT and DELETE keys. Last press the return key 25 times, you should see a reversed. Now enter

line 10 by pressing the return key. When the program is run you will not be able to get a listing, and line 10 will be invisible. Actually line 10 is listed, and then deleted. If you look close, you will see it blink on the screen for a very short time. To overcome this type of protection, line 10 can be deleted, providing it has no information that is required for the program to operate. To make line 10 invisible, the line has to be retyped, leaving out the deletes. NOTE-THIS TYPE OF PROTECTION CAN BE ANYWHERE IN YOUR PROGRAM.

10 POKE808,225

20 ?"ANYTHING"

30 GOTO 20

EXPLANATION - After the program above has been run, and not before, you will be unable to stop it. This is because, the poke in line 10, disables the RUN/STOP and RESTORE keys. Also when you attempt to get a program listing it will be unreadable.

DISK PROTECTION

LOOPING DIRECTORY:

The directory of a disk is on track 18, all disk drive functions start with track 18. You will need a sector editor to view the data on a disk, sector editors can be found in the public domain, free of charge. The sector editor is a program that is designed to view and modify the track and sectors of a disk. Disk doctor is a sector editor that is in the public domain, and there are many more commercial sector editors on the market. I would suggest that you acquire this type of software, if you plan to do much work with the disk drive. Assuming you have a sector editor, load and run it. Insert a disk that has programs on it, please be sure that you have a backup of this disk, in the event that you make a mistake, and destroy it. You should see a prompt, requesting which track and sector you want to view. Enter track 18 sector 1, you should see a display similar to the one on page 30. Enter track 18 Sector 1, you should see a display similar to the one on page 40. The first two bytes 12 and 04, they are the pointers to the next block and sector. The 12 is the hexadecimal equivalent of the decimal number 18, which is track 18, and 04 is sector 04. These first two bytes are the pointers to the next sector that has files stored on it. If this was the last sector of the directory file the two bytes would be the hexadecimal 00 FF, meaning that this is the end of the director files. The number of sectors that are used for the directory depends on how many programs are on the disk. To put a loop in the directory we will change the first two bytes on the last sector of the directory file from 00 FF to 12 01 (hexidecimal for 18 01). Having done this the disk drive will search in vain for the last sector on the directory file. This will not prevent programs on the disk from loading and running, only from listing the directory.

DIRECTORY FIX:

To fix a directory that has a loop, find where the first two bytes point to track 18 sector 01. Then change them to the hexadecimal numbers 00 FF. Now the directory is back to normal.

UNLISTABLE DIRECTORY:

Load and run your sector editor, go to track 18, sector 00. Each byte is two numbers long (example) 12, now count the bytes on track 18 sector 00. Bytes 144 through 161 contain the name of the disk, change the first six bytes to 14, 14, 14, 00, 00, 00. Now the directory is unlistable, change the six bytes back to what they were before to have a directory that lists.

DISK ERRORS:

One of the most used forms of protection consist of putting intentional errors on the disk. A list and description of these errors is below.

1. 20 (Read error, the disk drive is unable to locate the data block header).
2. 21 (Read error, the disk drive is unable to find a syn mark on the track.)
3. 22 (Read error, the disk drive has been told to read or verify a data block that was not written correctly.)
4. 23 (Read error, the disk drive has found a checksum error in a data block.)
5. 27 (Read error, the disk drive has found a checksum error in header.)
6. 29 (Disk ID MISMATCH)

It works like this, first the program on the disk has a routine that checks for a particular error condition on the disk. Then the read/write head of the disk drive is sent out to a specified track and sector to check if this error is present. If the error is not present the program is directed to crash. The protection routine below can be used in your software.

10000 T=1:S=1

10001 OPEN 15,8,15,"IO":REM OPEN ERROR CHANNEL AND INITIALIZE DRIVE

10002 OPEN 8,8,8:REM OPEN CHANNEL FOR DATA

10003 PRINT#15, "U1?:8,0";T;S:REM BLOCK READ* USING CHANNEL 8, DRIVE 0

10004 INPUT#15,A\$,B\$,C\$,D\$: REM READ ERROR CHANNEL

10005 CLOSE8:CLOSE15:REM CLOSE ALL FILES

10006 IF VAL(A\$)<>23 THEN 64760:REM IF ERROR VALUE NOT PRESENT THEN RESET

10007 RETURN:REM RETURN TO START OF PROGRAM

Amend this subroutine to your program. Then use a gosub statement to check for an error on the disk. Next you have to create an error 23 on the disk using a routine out of the book INSIDE THE COMMODORE DOS by RICHARD IMMERS AND GERALD G. NEUFELD. If this is not available MEGASOFT LTD. markets a program that will put any type of error on your disk.

TRACK 18 SECTOR 0

```

00 :12 01 41 00 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : A  00 00 00
10 :15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : 00 00 00 00
20 :15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : 00 00 00 00
30 :15 FF FF 1F 15 FF FF 1F 11 D7 5F 1F 00 00 00 00 : 00 00 00 00
40 :00 00 00 00 00 00 00 00 10 EC FF 07 00 00 00 00 : 00 00 00 00
50 :00 00 00 00 12 BF FF 07 13 FF FF 07 13 FF FF 07 : 00 00 00 00
60 :13 FF FF 07 12 FF FF 03 12 FF FF 03 12 FF FF 03 : 00 00 00 00
70 :12 FF FF 03 12 FF FF 03 12 FF FF 03 11 FF FF 01 : 00 00 00 00
80 :11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01 : 00 00 00 00
90 :31 35 34 31 54 45 53 54 2F 44 45 4D 4F A0 A0 A0 : 1541TEST/DEMO
A0 :A0 A0 5A 58 A0 32 41 A0 A0 A0 A0 00 00 00 00 : ZX 2A
B0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
C0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
D0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
E0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
F0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :

```

TRACK 18 SECTOR 1

```

00 :12 04 B2 11 00 48 4F 57 20 54 4F 20 55 53 45 A0 : 00 HOW TO USE
10 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 0D 00 :
20 :00 00 B2 11 03 48 4F 57 20 50 41 52 54 20 54 57 : 00 HOW PART TW
30 :4F A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 05 00 :
40 :00 00 B2 11 09 56 49 43 2D 32 30 20 57 45 44 47 : 00 VIC-20 WEDG
50 :45 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 04 00 : E
60 :00 00 B2 13 00 43 2D 36 34 20 57 45 44 47 45 A0 : 00 C-64 WEDGE
70 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00 :
80 :00 00 B2 13 01 44 4F 53 20 35 2E 31 A0 A0 A0 A0 : 00 DOS 5.1
90 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 04 00 :
A0 :00 00 B2 13 03 43 4F 50 59 2F 41 4C 4C A0 A0 A0 : 00 COPY/ALL
B0 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 08 00 :
C0 :00 00 B2 13 09 50 52 49 4E 54 45 52 20 54 45 53 : 00 PRINTER TES
D0 :54 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 09 00 : T
E0 :00 00 B2 10 00 44 49 53 4B 20 41 44 44 52 20 43 : 00 DISK ADDR C
F0 :4B 41 4E 47 45 00 00 00 00 00 00 00 00 04 00 : HANGE

```

TRACK 18 SECTOR 4

```

00 :00 FF B2 10 01 44 49 52 A0 A0 A0 A0 A0 A0 A0 : 00 DIR
10 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 04 00 :
20 :00 00 B2 10 03 56 49 45 57 20 42 41 4D A0 A0 A0 : 00 VIEW BAM
30 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 06 00 :
40 :00 00 B2 10 07 43 48 45 43 4B 20 44 49 53 4B A0 : 00 CHECK DISK
50 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 04 00 :
60 :00 00 B2 10 0F 44 49 53 50 4C 41 59 20 54 26 53 : 00 DISPLAY T&S
70 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 0E 00 :
80 :00 00 B2 14 02 50 45 52 46 4F 52 4D 41 4E 43 45 : 00 PERFORMANCE
90 :20 54 45 53 54 00 00 00 00 00 00 00 00 00 09 00 : TEST
A0 :00 00 B2 14 07 53 45 51 55 45 4E 54 49 41 4C 20 : 00 SEQUENTIAL
B0 :46 49 4C 45 A0 00 00 00 00 00 00 00 00 00 05 00 : FILE
C0 :00 00 B2 0F 01 52 41 4E 44 4F 4D 20 46 49 4C 45 : 00 RANDOM FILE
D0 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 0D 00 :
E0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
F0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :






```




























ASCII CODES

ASCII VALUE	CHR\$ DISPLAY	ASCII VALUE	CHR\$ DISPLAY
0		30	GREEN
1		31	BLUE
2		32	SPACE
3		33	!
4		34	"
5	WHITE	35	#
6		36	\$
7		37	%
8	DISABLE	38	&
	SHIFT COMMODORE	39	'
9	ENABLE	40	(
	SHIFT COMMODORE	41)
10		42	*
11		43	+
12		44	,
13	RETURN	45	-
14	LOWERCASE	46	.
15		47	/
16		48	0
17	CURSOR DOWN	49	1
18	REVERSE ON	50	2
19	HOME	51	3
20	DELETE	52	4
21		53	5
22		54	6
23		55	7
24		56	8
25		57	9
26		58	:
27		59	;
28	RED	60	<
29	CURSOR RIGHT	61	=
		62	>
		63	?

ASCII VALUE	CHR\$ DISPLAY	ASCII VALUE	CHR\$ DISPLAY
64	@	102	
65	A	103	
66	B	104	
67	C	105	
68	D	106	
69	E	107	
70	F	108	
71	G	109	
72	H	110	
73	I	111	
74	J	112	
75	K	113	
76	L	114	
77	M	115	
78	N	116	
79	O	117	
80	P	118	
81	Q	119	
82	R	120	
83	S	121	
84	T	122	
85	U	123	
86	V	124	
87	W	125	
88	X	126	
89	Y	127	
90	Z	128	
91	[129	
92	\	130	
93]	131	
94	^	132	
95	_	133	
96	`	134	
97	a	135	
98	b	136	
99	c	137	
100	d	138	
101	e	139	








f1
f3
f5
f7
f2
f4
f6

ASCII VALUE	CHR\$ DISPLAY
140	f8
141	SHIFTED RETURN
142	UPPERCASE
143	
144	BLACK
145	CURSOR UP
146	REVERSE OFF
147	CLEAR SCREEN
148	INSERT
149	BROWN
150	LIGHT RED
151	GRAY 1
152	GRAY 2
153	LIGHT GREEN
154	LIGHT BLUE
155	GRAY 3
156	PURPLE
157	CURSOR LEFT
158	YELLOW
159	CYAN
160	SPACE
161	
162	
163	
164	
165	

ASCII VALUE	CHR\$ DISPLAY
166	
167	
168	
169	
170	
171	
172	
173	
174	
175	
176	
177	
178	
179	
180	
181	
182	
183	
184	
185	
186	
187	
188	
189	
190	
191	

CODES	192-223	SAME AS	96-127
CODES	224-254	SAME AS	160-190
CODE	255	SAME AS	128

SCREEN CODES

POKE VALUE	SET 1	SET 2	POKE VALUE	SET 1	SET 2
0	@	@	35	#	
1	A	a	36	\$	
2	B	b	37	%	
3	C	c	38	&	
4	D	d	39	'	
5	E	e	40	(
6	F	f	41)	
7	G	g	42	*	
8	H	h	43	+	
9	I	i	44	,	
10	J	j	45	-	
11	K	k	46	.	
12	L	l	47	/	
13	M	m	48	0	
14	N	n	49	1	
15	O	o	50	2	
16	P	p	51	3	
17	Q	q	52	4	
18	R	r	53	5	
19	S	s	54	6	
20	T	t	55	7	
21	U	u	56	8	
22	V	v	57	9	
23	W	w	58	:	
24	X	x	59	;	
25	Y	y	60	<	
26	Z	z	61	=	
27			62	>	>
28	£		63	?	?
29			64		
30	↑		65		A
31	←		66		B
32	SPACE		67		C
33	!		68		D
34	"		69		E

POKE
VALUE

SET 1

SET 2

70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98



F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z



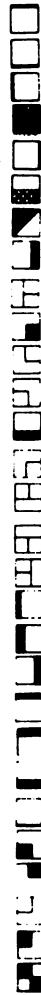
SPACE

POKE
VALUE

SET 1

SET 2

99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127



SCREEN DISPLAY	KEY(S) TO PRESS	FUNCTION
	CLR/home	Cursor home
	CRSR	Cursor down
	CRSR	Cursor right
	CTRL-9	Reverse on
	Shift-CLR/home	Clear screen
	Shift-CRSR	Cursor up
	Shift-CRSR	Cursor left
	CTRL-0	Reverse off
SCREEN DISPLAY	KEY(S) TO PRESS	COLORS
	CTRL-1	BLACK
	CTRL-2	WHITE
	CTRL-3	RED
	CTRL-4	CYAN
	CTRL-5	PURPLE
	CTRL-6	GREEN
	CTRL-7	BLUE
	CTRL-8	YELLOW
	- 1	BROWN
	- 2	ORANGE
	- 3	LIGHT RED
	- 4	DARK GRAY
	- 5	MEDIUM GRAY
	- 6	LIGHT GREEN
	- 7	LIGHT BLUE
	- 8	LIGHT GRAY

® Copyright 1984 by G.L. Stacey
ALL RIGHTS RESERVED

Published by: *MegaSoft, Ltd.*
P.O. Box 1080, Battle Ground, WA 98604